Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Фізико-Технічний інститут

КРИПТОГРАФІЯ
КОМП’ЮТЕРНИЙ ПРАКТИКУМ №1

Виконала: Студентка 3-го курсу
Групи ФБ-93
Пономаренко Олександра Сергіївна

Київ 2021

Експериментальна оцінка ентропії на символ джерела відкритого тексту

Мета:

Засвоєння понять ентропії на символ джерела та його надлишковості, вивчення та порівняння різних моделей джерела відкритого тексту для наближеного визначення ентропії, набуття практичних навичок щодо оцінки ентропії на символ джерела.

Завдання до виконання:

0. Уважно прочитати методичні вказівки до виконання комп'ютерного практикуму.

1. Написати програми для підрахунку частот букв і частот біграм в тексті без пробілів (з пробілами), а також підрахунку H_1, H_2 та H_3 за безпосереднім означенням (і для тексту із пробілами відповідні ентропії).
2. За допомогою програми CoolPinkProgram оцінити значення $H^{(10)}$, $H^{(20)}$, $H^{(30)}$.
3. Використовуючи отримані значення ентропії, оцінити надлишковість російської мови в різних моделях джерела.

Виконання роботи:

Застосовані функції:

1)для читання з файлу;
2)для запису в файл;

```cpp
#include <iostream>
#include <string>
#include <iomanip>
#include <algorithm>
#include <clocale>
#include <fstream>
#include <windows.h>
#include <math.h>

using namespace std;

//читання з файлу
string inputf(ifstream& f, char str[40]) {
    string a;
    f.open(str);
    if (f.fail()) {
        cout << "\n Fail to open the file";
        exit(1);
    }
    string extra = "";
    while (getline(f, a)) {
        extra += a;
    }
    f >> extra;
    f.close();
    return extra;
}

//запис у файл
void outputf(ofstream& f, string a, char str[40]) {
    f.open(str);
    if (f.fail()) {
        cout << "\n Fail to open the file";
        exit(1);
    }
    f << a;
    f.close();
}
```

3)-для знаходження матриці частоти біграм (з перетином)
-для запису відповідної матриці
-для знаходження відповідної ентропії

```cpp
double withIntersection(string** a_bigram, int lengthOfArray, string txt_filtered, int lengthOfText, ofstream& f1, char str[40], string alphabet) {
    string bigram = "";
    char sym1;
    char sym2;
    //цикл на заповнення біграм алфавіту
    for (int r = 0; r < lengthOfArray; ++r) {
        for (int c = 0; c < lengthOfArray; ++c) {
            sym1 = alphabet[r];
            sym2 = alphabet[c];
            string bi1(1, sym1);
            string bi2(1, sym2);
            bigram = bi1 + bi2;
            a_bigram[r][c] = bigram;
        }
    }
    int** counter = new int* [lengthOfArray];
    string** sa_frequencyOfBi = new string * [lengthOfArray];
    double** a_frequencyOfBi = new double* [lengthOfArray];
    for (int i = 0; i < lengthOfArray; i++) {
        counter[i] = new int[lengthOfArray];
        sa_frequencyOfBi[i] = new string[lengthOfArray];
        a_frequencyOfBi[i] = new double[lengthOfArray];
    }
    for (int i = 0; i < lengthOfArray; i++) {
        for (int j = 0; j < lengthOfArray; j++) {
            counter[i][j] = 0;
        }
    }

    int j1, j2;
    //цикл на підрахунок біграм (так як крок i=i+1 - біграми з перетином)
    for (int i = 0; i < lengthOfText; i++) {
        j1 = alphabet.find(txt_filtered[i]);
        j2 = alphabet.find(txt_filtered[i + 1]);
        counter[j1][j2]++;
    }
    double frequencyOfBi;
    //цикл на підрахунок частоти кожної біграми
    for (int r = 0; r < lengthOfArray; ++r) {
        for (int c = 0; c < lengthOfArray; ++c) {
            frequencyOfBi = (counter[r][c] * 1.0) / (lengthOfText - 1);
            //заповнення додаткової стрінгової матриці, яку ми запишемо у файл
            sa_frequencyOfBi[r][c] = a_bigram[r][c] + ": " + to_string(frequencyOfBi) + " ";
            a_frequencyOfBi[r][c] = frequencyOfBi;
        }
    }
    //запис матриці у файл
    f1.open(str);
    if (f1.fail()) {
        cout << "\n Fail to open the file";
        exit(1);
    }
    for (int r = 0; r < lengthOfArray; ++r) {
        f1 << endl;
        for (int c = 0; c < lengthOfArray; ++c) {
            f1 << setw(10) << sa_frequencyOfBi[r][c];
        }
    }

    f1.close();
    double entropy = 0;
    double logarifm;
    //цикл на знаходження ентропії
    for (int r = 0; r < lengthOfArray; ++r) {
        for (int c = 0; c < lengthOfArray; ++c) {
            logarifm = log10(a_frequencyOfBi[r][c]) / log10(2);
            //у тому випадку, коли частота біграми=0, логаріфм дорівнюватиме нескінченності
            if (logarifm == -INFINITY) {
                logarifm = 0;//запишемо через нуль для зручності(адже ми все одно будемо множити на нуль)
            }
            logarifm = a_frequencyOfBi[r][c] * logarifm;
            entropy = entropy + logarifm;
        }
    }
    entropy = entropy * (-1);
    return entropy;
}
```

4)-для знаходження матриці частоти біграм (без перетину)
-для запису відповідної матриці
-для знаходження відповідної ентропії

```
118 double matrix(string** a_bigram, int lengthOfArray, string txt_filtered, int lengthOfText, ofstream& f1, char str[40], string alphabet)
119     string bigram = "";
120     char sym1;
121     char sym2;
122     //цикл на заповнення біграм алфавіту
123     for (int r = 0; r < lengthOfArray; ++r) {
124         for (int c = 0; c < lengthOfArray; ++c) {
125             sym1 = alphabet[r];
126             sym2 = alphabet[c];
127             string bi1(1, sym1);
128             string bi2(1, sym2);
129             bigram = bi1 + bi2;
130             a_bigram[r][c] = bigram;
131         }
132     }
133     int** counter = new int* [lengthOfArray];
134     string** sa_frequencyOfBi = new string * [lengthOfArray];
135     double** a_frequencyOfBi = new double* [lengthOfArray];
136     for (int i = 0; i < lengthOfArray; i++) {
137         counter[i] = new int[lengthOfArray];
138         sa_frequencyOfBi[i] = new string[lengthOfArray];
139         a_frequencyOfBi[i] = new double[lengthOfArray];
140     }
141     for (int i = 0; i < lengthOfArray; i++) {
142         for (int j = 0; j < lengthOfArray; j++) {
143             counter[i][j] = 0;
144         }
145     }
146
147     int j1, j2;
148     //цикл на підрахунок біграм (так як крок i=i+2 - біграми без перетину)
149     for (int i = 0; i < lengthOfText; i = i + 2) {
150         j1 = alphabet.find(txt_filtered[i]);
151         j2 = alphabet.find(txt_filtered[i + 1]);
152         counter[j1][j2]++;
153     }
154     double frequencyOfBi;
155     //цикл на підрахунок частоти кожної біграми
156     for (int r = 0; r < lengthOfArray; ++r) {
157         for (int c = 0; c < lengthOfArray; ++c) {
158             frequencyOfBi = (counter[r][c] * 1.0) / (lengthOfText / 2);
159             sa_frequencyOfBi[r][c] = a_bigram[r][c] + ": " + to_string(frequencyOfBi) + " ";
160             a_frequencyOfBi[r][c] = frequencyOfBi;
161         }
162     }
```

```
163     //запис матриці у файл
164     f1.open(str);
165     if (f1.fail()) {
166         cout << "\n Fail to open the file";
167         exit(1);
168     }
169     for (int r = 0; r < lengthOfArray; ++r) {
170         f1 << endl;
171         for (int c = 0; c < lengthOfArray; ++c) {
172             f1 << setw(10) << sa_frequencyOfBi[r][c];
173         }
174     }
175     f1.close();
176
177     double entropy = 0;
178     double logarifm;
179     //цикл на знаходження ентропії
180     for (int r = 0; r < lengthOfArray; ++r) {
181         for (int c = 0; c < lengthOfArray; ++c) {
182             logarifm = log10(a_frequencyOfBi[r][c]) / log10(2);
183             if (logarifm == -INFINITY) {
184                 logarifm = 0;
185             }
186             logarifm = a_frequencyOfBi[r][c] * logarifm;
187             entropy = entropy + logarifm;
188         }
189     }
190     entropy = entropy * (-1);
191     return entropy;
192 }
```

5)-для знаходження ентропії із частотами літер

```
194 double entropy(double* a, int lengthOfArray) {
195     double entropy = 0;
196     double logarifm;
197     for (int i = 0; i < lengthOfArray; i++) {
198         logarifm = log10(a[i]) / log10(2);
199         logarifm = a[i] * logarifm;
200         entropy = entropy + logarifm;
201     }
202     entropy = entropy * (-1);
203     return entropy;
204 }
```

6)-для підрахунку усіх літер
-для підрахунку частоти кожної літери
-для сортування частот

```
206 string sorted_txt(int lengthOfArray, string txt_filtered, int* a_quantityOfSym, int lengthOfText, string* as_frequencyOfSym, double* a_frequencyOfSym, string alphabet) {
207     double frequencyOfSym;
208     char symbol;
209     string frequencyWithSym = "";
210     string frequencySorted = "";
211     //цикл на підрахунок усіх літер в тексті(від а до я)
212     for (int i = 0; i < lengthOfArray; i++) {
213         symbol = alphabet[i];
214         const auto count_symbol = count(txt_filtered.cbegin(), txt_filtered.cend(), symbol);
215         a_quantityOfSym[i] = count_symbol;//запис кі-сті літер у масив
216     }
217     //цикл на підрахунок частоти кожної літери
218     for (int i = 0; i < lengthOfArray; i++) {
219         frequencyOfSym = (a_quantityOfSym[i] * 1.0) / lengthOfText;
220         a_frequencyOfSym[i] = frequencyOfSym;//запис частоти кожної літери у масив
221         as_frequencyOfSym[i] = to_string(frequencyOfSym) + " (" + alphabet[i] + ")";//запис частоти і відповідної літери у масив
222     }
223     for (int i = 0; i < lengthOfArray; i++) {
224         frequencyWithSym = frequencyWithSym + as_frequencyOfSym[i] + "\n";
225     }
226     //цикл на сортування масиву із частотами
227     for (int i = 0; i < lengthOfArray-1; ++i) {
228         int bigger = i;
229         for (int j = i + 1; j < lengthOfArray; ++j) {
230             if (a_frequencyOfSym[j] > a_frequencyOfSym[bigger]) {
231                 bigger = j;
232             }
233         }
234         swap(a_frequencyOfSym[i], a_frequencyOfSym[bigger]);
235     }
236     for (int i = 0; i < lengthOfArray; i++) {
237         frequencySorted = frequencySorted + to_string(a_frequencyOfSym[i]) + "\n";
238     }
239     txt_filtered = frequencyWithSym + "\n" + "Sorted:" + "\n" + frequencySorted;
240     return txt_filtered;
241 }
```

Відповідні текстові файли:

- **vim_whs.txt** - фільтрований текст без пробілів; **intersection_matrix.txt** - матриця частот біграм (з перетином); **matrix.txt** - матриця частот біграм (без перетину); **sorted.txt** - відсортовані частоти літер. Ці всі файли знаходяться у папці **without_spaces/**.
- **vim_ws.txt** - фільтрований текст із пробілами; **intersection_matrix_ws.txt** - матриця частот біграм (з перетином); **matrix_ws.txt** - матриця частот біграм (без перетину); **sorted_ws.txt** - відсортовані частоти літер. Ці всі файли знаходяться у папці **with_spaces/**.

Далі перейдемо до мейну:

I. Текст без пробілів:

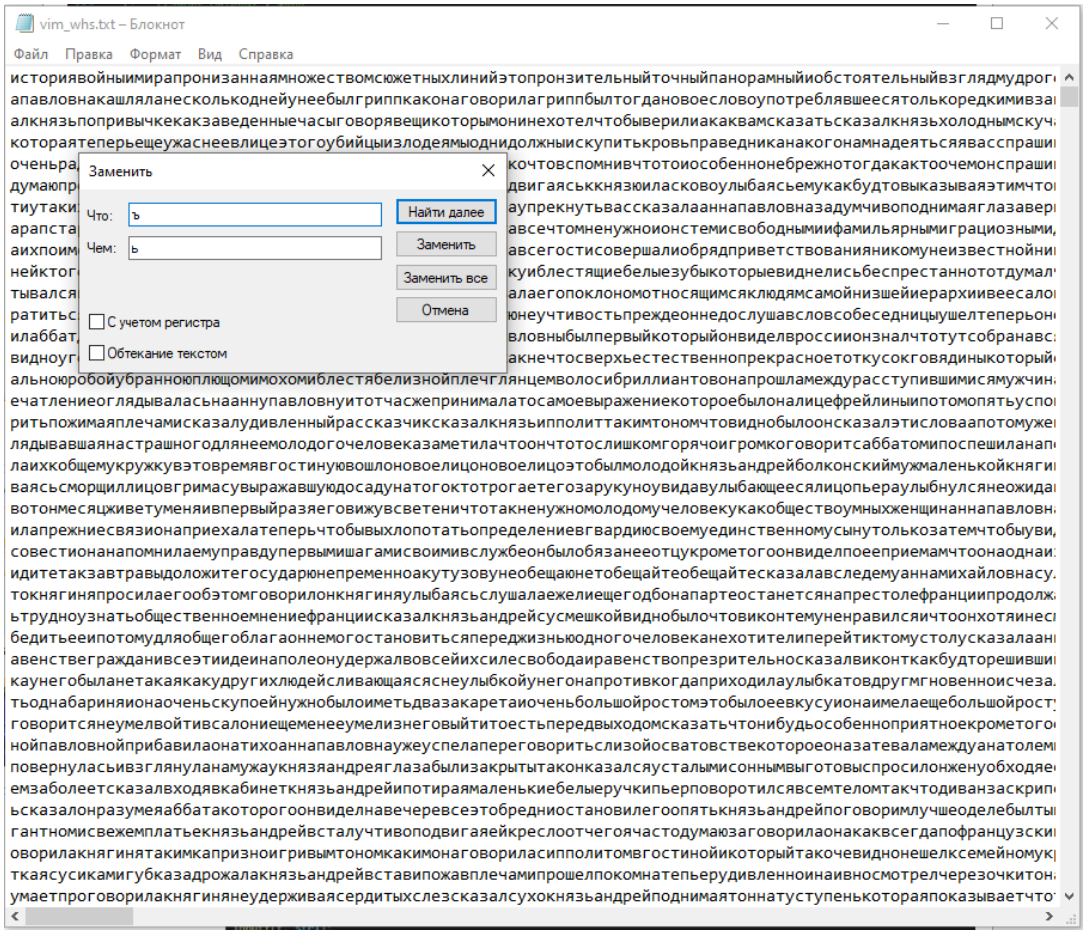
1)фільтрація тексту

```cpp
    setlocale(LC_ALL, "Russian");
    string txt_before, txt_after;
    string txt_sorted, txt_filtered;
    ifstream f;
    ofstream f1;
    char str1[40], str2[40];
    cout << "Enter the root to .txt file: ";
    cin >> str1;
    txt_before = inputf(f, str1);
    txt_after = txt_before;

    //фільтрація тексту
    txt_after.erase(remove_if(txt_after.begin(), txt_after.end(), [](char c) { return !isalpha((unsigned char)c); }), txt_after.end());
    char chars[] = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ";
    for (unsigned int i = 0; i < strlen(chars); ++i) {
        txt_after.erase(std::remove(txt_after.begin(), txt_after.end(), chars[i]), txt_after.end());
    }
    //верхній регістр->нижній регістр
    transform(txt_after.begin(), txt_after.end(), txt_after.begin(), tolower);

    cout << "New .txt is ready. Enter the root to it: ";
    cin >> str2;
    outputf(f1, txt_after, str2);
```

Залишилося замінити усі літери ‘ё’ на ‘е’, та ‘ъ’ на ‘ь’. Зробити це можна у звичайному блокноті



2) Проходження по ф-ціям, запис у файли та підрахунок усіх ентропій

```
293     string alphabet;
294     int lengthOfText;
295     int lengthOfArray;
296     //зчитуємо змінений файл
297     cout << "Enter the name of filtered txt(without spaces): ";
298     cin >> str1;
299     txt_filtered = inputf(f, str1);
300     alphabet = "абвгдежзийклмнопрстуфхцчшщьыэюя";
301     lengthOfArray = alphabet.length();
302     lengthOfText = txt_filtered.length();
303     cout << "Length of new txt: " << lengthOfText << " symbols" << endl;
304     string* as_frequencyOfSym = new string[lengthOfArray];
305     int* a_quantityOfSym = new int[lengthOfArray];
306     double* a_frequencyOfSym = new double[lengthOfArray];
307     txt_sorted = sorted_txt(lengthOfArray, txt_filtered, a_quantityOfSym, lengthOfText, as_frequencyOfSym, a_frequencyOfSym, alphabet);
308     cout << "The frequency of symbols in the new .txt has been sorted. Enter the root to it: ";
309     cin >> str2;
310     outputf(f1, txt_sorted, str2);
311     cout << "Entropy(symbol frequencies): " << entropy(a_frequencyOfSym, lengthOfArray) << endl;
312
313     string** a_bigram = new string*[lengthOfArray];
314     for (int i = 0; i < lengthOfArray; i++) {
315         a_bigram[i] = new string[lengthOfArray];
316     }
317     cout << "The frequency of bigrams(with intersection) has been written. Enter the root to it: ";
318     cin >> str2;
319     cout << "Entropy(bigram frequencies with intersection): " << withIntersection(a_bigram, lengthOfArray, txt_filtered, lengthOfText, f1, str2, alphabet) << endl;
320
321     cout << "The frequency of bigrams has been written. Enter the root to it: ";
322     cin >> str2;
323     cout << "Entropy(bigram frequencies): " << matrix(a_bigram, lengthOfArray, txt_filtered, lengthOfText, f1, str2, alphabet) << endl;
```

<u>Знайдені значення ентропії:</u>

-із частотою літер

```
Length of new txt: 2302687 symbols
Entropy(symbol frequencies): 4.45931
```

-із частотою біграм (з перетином та без)

```
Entropy(bigram frequencies with intersection): 4.13845
Entropy(bigram frequencies): 4.13767
```

II. Текст із пробілами:

1)фільтрація тексту

```
//фільтрація тексту(щоб залишилися пробіли)
for (int i = 0; i < txt_after.length(); i++) {
    //cout << a[i] << endl;
    if (txt_after[i] == ' ') {
        txt_after.replace(i, 1, "q");
    }
}
txt_after.erase(remove_if(txt_after.begin(), txt_after.end(), [](char c) { return !isalpha((unsigned char)c); }), txt_after.end());
for (int i = 0; i < txt_after.length(); i++) {
    //cout << a[i] << endl;
    if (txt_after[i] == 'q') {
        txt_after.replace(i, 1, " ");
    }
}
char chars[] = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ";
for (unsigned int i = 0; i < strlen(chars); ++i) {
    txt_after.erase(remove(txt_after.begin(), txt_after.end(), chars[i]), txt_after.end());
}
for (int i = txt_after.size() - 1; i >= 0; i--) {
    if (txt_after[i] == ' ' && txt_after[i] == txt_after[i - 1]) {
        txt_after.erase(txt_after.begin() + i);
    }
}
transform(txt_after.begin(), txt_after.end(), txt_after.begin(), tolower);
```

Також замінимо усі літери ‘ё’ на ‘е’, та ‘ъ’ на ‘ь’.

2) Проходження по ф-ціям, запис у файли та підрахунок усіх ентропій

```
cout << "Enter the name of filtered txt(with spaces): ";
cin >> str1;
txt_filtered = inputf(f, str1);
alphabet = "абвгдежзийклмнопрстуфхцчшщыьэюя ";
lengthOfArray = alphabet.length();
lengthOfText = txt_filtered.length();
cout << "Length of new txt: " << lengthOfText << " symbols" << endl;
string* as_frequencyOfSym2 = new string[lengthOfArray];
int* a_quantityOfSym2 = new int[lengthOfArray];
double* a_frequencyOfSym2 = new double[lengthOfArray];
txt_sorted = sorted_txt(lengthOfArray, txt_filtered, a_quantityOfSym2, lengthOfText, as_frequencyOfSym2, a_frequencyOfSym2, alphabet);
cout << "The frequency of symbols in the new .txt has been sorted. Enter the root to it: ";
cin >> str2;
outputf(f1, txt_sorted, str2);
cout << "Entropy(symbol frequencies): " << entropy(a_frequencyOfSym2, lengthOfArray) << endl;

string** a_bigram2 = new string * [lengthOfArray];
for (int i = 0; i < lengthOfArray; i++) {
    a_bigram2[i] = new string[lengthOfArray];
}
cout << "The frequency of bigrams(with intersection) has been written. Enter the root to it: ";
cin >> str2;
cout << "Entropy(bigram frequencies with intersection): " << withIntersection(a_bigram2, lengthOfArray, txt_filtered, lengthOfText, f1, str2, alphabet) << endl;

cout << "The frequency of bigrams has been written. Enter the root to it: ";
cin >> str2;
cout << "Entropy(bigram frequencies): " << matrix(a_bigram2, lengthOfArray, txt_filtered, lengthOfText, f1, str2, alphabet) << endl;
```

Знайдені значення ентропії:

-із частотою літер

```
Length of new txt: 2747315 symbols
The frequency of symbols in the new .txt has been sorted. Enter the root to it: sorted_ws.txt
Entropy(symbol frequencies): 4.37631
```

-із частотою біграм (з перетином та без)

```
Entropy(bigram frequencies with intersection): 3.96615
Entropy(bigram frequencies): 3.96641
```

Хід роботи та опис труднощів:

Для написання даного комп'ютерного практикуму використовувала мову c++. За основу лабораторної роботи обрала книгу "Война и Мир". Фільтрація тексту на c++ проходить легко й без проблем, за допомогою вбудованної ф-кції isalpha(), яка перевіряє текст на наявність символів, окрім літер (якщо таких символів немає - повертає 1).

Під час зчитування тексту книги та запису фільтрованого тексту з'явилася проблема із кодуванням. Замість російських літер програма зчитувала та записувала незрозумілі позначки. Але це легко вирішилося переведенням кодування на ANSI.

Сортування із частотами літер я зробила за допомогою сортування масиву вибором. Реалізація нескладна через використання вбудованної ф-ції swap(), яка міняє місцями два обрані елементи масиву.

Для обрахунків ентропії використовуємо формулу:

$$H(Z) = -\sum_{i=1}^{n} p_i \log p_i \ .$$

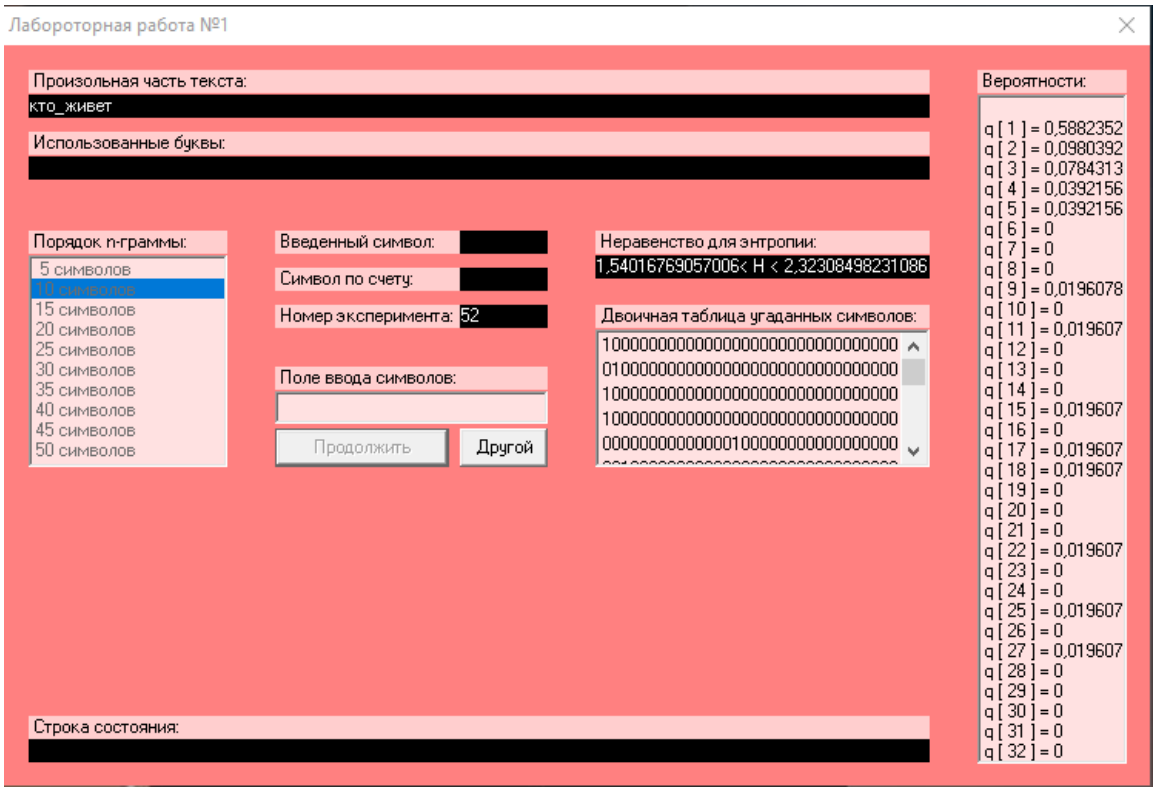
замінюючи імовірність на відповідну частоту. Так як зазвичай основою логарифму беруть двійку, потрібно зробити перехід від основи 10 до основи 2. Робиться це за формулою $\log_{10}(x)/ \log_{10}2$.

Також під час обрахунків ентропії біграм зустрінемося з випадком, коли значення p=0, а отже логарифм буде дорівнювати -нескінченності. А нуль помножити на -нескінченність = невизначеність. На практиці, використовуючи правило Лопіталя, можна довести, що це значення все ж таки дорівнюватиме нулю, але програма буде видавати nan. Щоб запобігти такої неприємної ситуації, я вирішила додати умову:

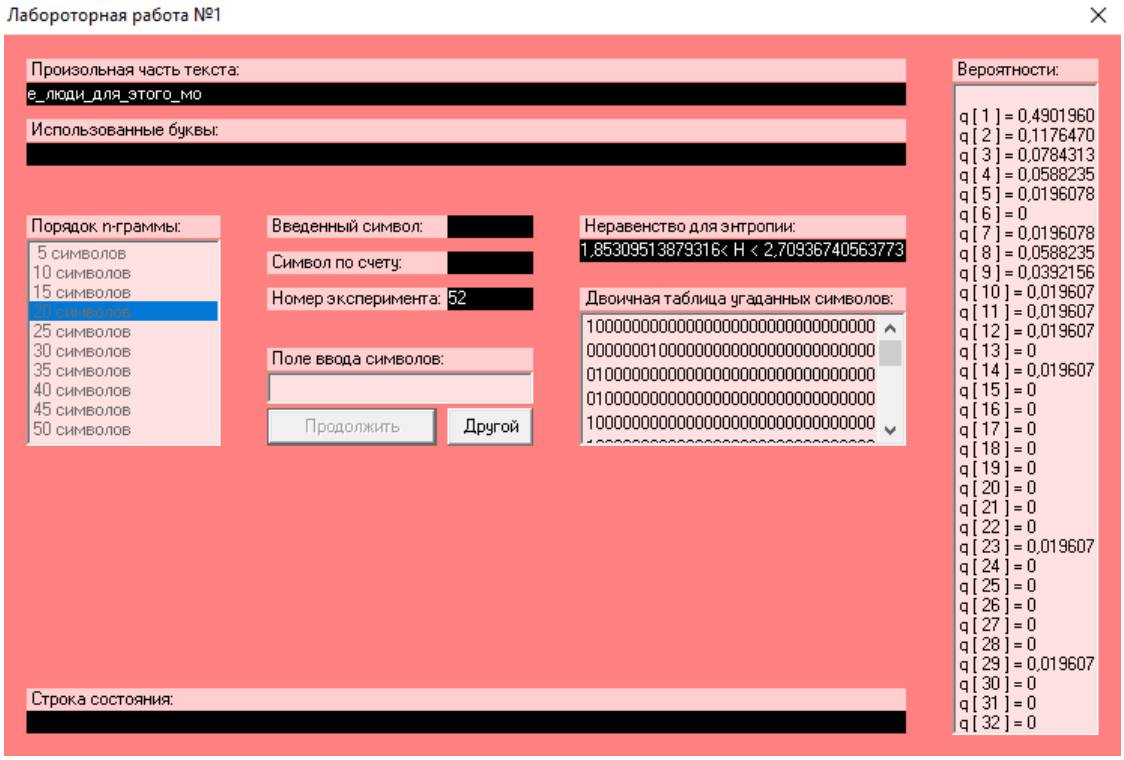
```
for (int r = 0; r < lengthOfArray; ++r) {
    for (int c = 0; c < lengthOfArray; ++c) {
        logarifm = log10(a_frequencyOfBi[r][c]) / log10(2);
        //у тому випадку, коли частота біграми=0, логаріфм дорівнюватиме нескінченності
        if (logarifm == -INFINITY) {
            logarifm = 0;//запишемо через нуль для зручності(адже ми все одно будемо множити на нуль)
        }
        logarifm = a_frequencyOfBi[r][c] * logarifm;
        entropy = entropy + logarifm;
    }
}
entropy = entropy * (-1);
```

2) За допомогою програми CoolPinkProgram оцінити значення $H^{(10)}$, $H^{(20)}$, $H^{(30)}$:

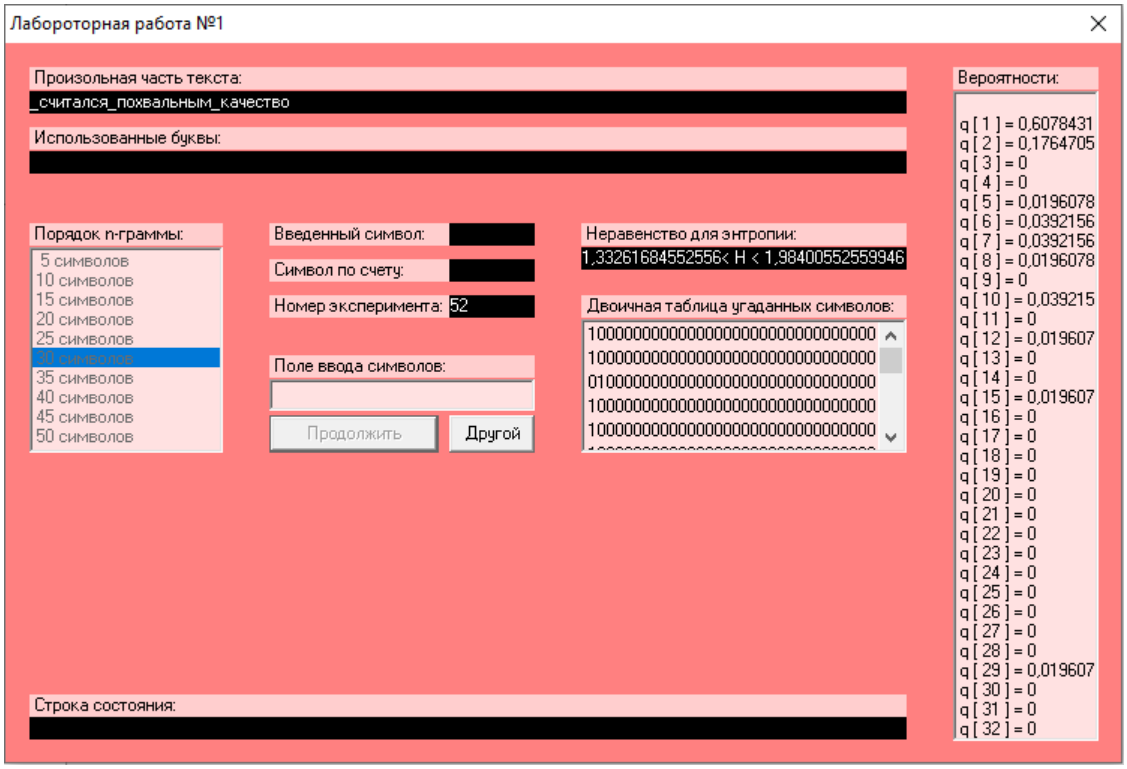
А) $H^{(10)}$



Б) $H^{(20)}$


Лабороторная работа №1
Произольная часть текста:
е_люди_для_этого_мо
Использованные буквы:
Порядок n-граммы:
5 символов
10 символов
15 символов
20 символов
25 символов
30 символов
35 символов
40 символов
45 символов
50 символов
Введенный символ:
Символ по счету:
Номер эксперимента: 52
Поле ввода символов:
Продолжить
Другой
Неравенство для энтропии:
1,85309513879316< H < 2,70936740563773
Двоичная таблица угаданных символов:
Вероятности:
q [1] = 0,4901960
q [2] = 0,1176470
q [3] = 0,0784313
q [4] = 0,0588235
q [5] = 0,0196078
q [6] = 0
q [7] = 0,0196078
q [8] = 0,0588235
q [9] = 0,0392156
q [10] = 0,019607
q [11] = 0,019607
q [12] = 0,019607
q [13] = 0
q [14] = 0,019607
q [15] = 0
q [16] = 0
q [17] = 0
q [18] = 0
q [19] = 0
q [20] = 0
q [21] = 0
q [22] = 0
q [23] = 0,019607
q [24] = 0
q [25] = 0
q [26] = 0
q [27] = 0
q [28] = 0
q [29] = 0,019607
q [30] = 0
q [31] = 0
q [32] = 0
Строка состояния:

В) $H^{(30)}$


Лабороторная работа №1
Произольная часть текста:
_считался_похвальным_качество
Использованные буквы:
Порядок n-граммы:
5 символов
10 символов
15 символов
20 символов
25 символов
30 символов
35 символов
40 символов
45 символов
50 символов
Введенный символ:
Символ по счету:
Номер эксперимента: 52
Поле ввода символов:
Продолжить
Другой
Неравенство для энтропии:
1,33261684552556< H < 1,98400552559946
Двоичная таблица угаданных символов:
Вероятности:
q [1] = 0,6078431
q [2] = 0,1764705
q [3] = 0
q [4] = 0
q [5] = 0,0196078
q [6] = 0,0392156
q [7] = 0,0392156
q [8] = 0,0196078
q [9] = 0
q [10] = 0,039215
q [11] = 0
q [12] = 0,019607
q [13] = 0
q [14] = 0
q [15] = 0,019607
q [16] = 0
q [17] = 0
q [18] = 0
q [19] = 0
q [20] = 0
q [21] = 0
q [22] = 0
q [23] = 0
q [24] = 0
q [25] = 0
q [26] = 0
q [27] = 0
q [28] = 0
q [29] = 0,019607
q [30] = 0
q [31] = 0
q [32] = 0
Строка состояния:

3) Використовуючи отримані значення ентропії, оцінити надлишковість російської мови

Надлишковість мови обраховується за такою формулою:

$$R = 1 - \frac{H_\infty}{H_0}$$

$H_0 = \log_2 32 = 5$

1)1-(1.5401/5)>$R^{(10)}$>1-(2.323/5)

0.6919> $R^{(10)}$>0.5354

2) 1-(1.853/5)>$R^{(20)}$>1-(2.7093/5)

0.6294> $R^{(20)}$>0.4581

3) 1-(1.3326/5)>$R^{(30)}$>1-(1.984/5)

0.7335> $R^{(30)}$>0.6032

Висновок:

За цю лабораторну роботу ми дізналися про поняття ентропії та обрахували практично різні її значення для обраного тексту(з пробілами і без). Також для обчислення біграм згадали як працювати із двумірними масивами та згадали методи сортування масивів(для масиву частот літер). Також корисними для закріплення були робота із файлами(зчитування/запис) та ознайомлення з github’ом.